Maillet Louis
Belaarous Yassine

# SAE Algo - À la conquête d'Hollywood

# Rapport

3.1 :

Dans le graphe des collaborations, les sommets représentent les acteurs/actrices, et une arête entre deux sommets indique qu'ils ont joué ensemble dans un film.

donc Pour deux acteurs/actrices donnés A et B, leurs collaborateurs communs sont les sommets qui sont adjacents à la fois à A et à B dans le graphe.

Soient E(A) et E(B) les ensembles de voisins de A et B.
L'ensemble des collaborateurs en commun est :
$E(A) \cap E(B)$

3.2

l'objectif de la méthode collaborateur_proche est de déterminer tous les acteurs situés à une distance ≤ k à partir d'un acteur donné v1 il s'agit donc d'un parcours en largeur

Voici les éléments qui prouvent qu'il s'agit d'un parcours en largeur :

Il explore les sommets par couches, c'est-à-dire d'abord ceux à distance 1, puis ceux à distance 2, etc jusqu'a la distance k, il utilise un ensemble qui permet de stocker les sommets deja passé

Donc ici nous utilisons bien un parcours en largeurs

Grâce à la fonction collaborateur_proche, qui renvoie tous les acteurs situés à une distance inférieure ou égale à k d'un acteur donné dans un graphe g, on peut déterminer si un acteur A se trouve exactement à distance k d'un acteur B en procédant de la manière suivante :

1. On utilise collaborateur_proche(g, B, k) pour obtenir l'ensemble X des acteurs situés à une distance inférieure ou égale à k de B.
2. On utilise ensuite collaborateur_proche(g, B, k-1) pour obtenir l'ensemble Y des acteurs situés à une distance inférieure ou égale à k-1 de B.
3. Si l'acteur A appartient à l'ensemble X mais n'appartient pas à l'ensemble Y, alors cela signifie que A est exactement à distance k de B.:

En résumé, un acteur A est à distance exacte k d'un acteur B s'il appartient à l'ensemble des collaborateurs proches de B à distance k, mais qu'il n'apparaît pas dans l'ensemble des collaborateurs proches de B à distance k - 1.

On peut réutiliser la méthode collaborateur_proche(g, A, 1) pour vérifier si l'acteur B se trouve à une distance de 1 de A. Si B n'apparaît pas dans l'ensemble obtenu, on augmente progressivement la distance et on appelle collaborateur_proche(g, A, 2), puis collaborateur_proche(g, A, 3), etc., jusqu'à ce que B soit présent dans l'ensemble retourné. La première valeur de k pour laquelle B est trouvé correspond à la distance minimale entre A et B.

La complexité est de $O(n^4)$ car notre algorithme collaborateur_proche est en $O(n^3)$, et comme nous faisons ici une boucle while, cela ajoute une boucle supplémentaire, ce qui donne une complexité en $O(n^4)$.

3.4

La notion de théorie des graphes corresponds a l'excentricité.

Elle représente la plus grande distance minimale entre un acteur et les autres dans le graphe.

Suite des questions faite en java

Chapitre 4

Pour améliorer nos complexités, nous pourrions éviter d'appeler plusieurs fois la méthode centraliteActeur, qui effectue un parcours en largeur à chaque appel. Cela devient rapidement lent lorsque le nombre de sommets est important, car ces appels sont faits pour chaque sommet dans des méthodes comme centreGraphe ou petiteFamille.

Pour cela, nous pourrions effectuer un seul parcours en largeur depuis chaque sommet et stocker les distances entre toutes les paires de sommets dans une matrice de distances. Cette matrice permettrait ensuite de récupérer directement les distances nécessaires pour calculer les centralités, sans avoir à recalculer les chemins à chaque fois. Cette optimisation réduirait fortement la redondance des calculs et améliorerait significativement les performances globales sur des graphes de grande taille.

Pour cette partie, nous ne sommes pas trop sûrs de notre théorie car nous n'avons pas réussi à la mettre en place, mais sur certains forums, ils proposaient que mettre en place une matrice est plus performant. C'est donc notre conclusion

voici les site que nous avons pu utilisé pour essayer d’optimiser :
https://igraph.org/c/html/0.10.15/igraph-Structural.html